8-2012

On XACML's adequacy to specify and to enforce HIPAA

Omas Chowdhury

Haining Chen

Jianwei Niu

Ninghui Li

Elisa Bertino
Purdue University, bertino@cs.purdue.edu

On XACML's Adequacy to Specify and to Enforce HIPAA

Omar Chowdhury
The University of Texas at San Antonio,
San Antonio, TX, USA.
ochowdhu@cs.utsa.edu

Haining Chen
Purdue University,
West Lafayette, IN, USA.
chen623@cs.purdue.edu

Jianwei Niu
The University of Texas at San Antonio,
San Antonio, TX, USA.
niu@cs.utsa.edu.

Ninghui Li, Elisa Bertino
Purdue University,
West Lafayette, IN, USA.
{ninghui, bertino}@cs.purdue.edu

Abstract

In the medical sphere, personal and medical information is collected, stored, and transmitted for various purposes, such as, continuity of care, rapid formulation of diagnoses, and billing. Many of these operations must comply with federal regulations like the Health Insurance Portability and Accountability Act (HIPAA). To this end, we need a specification language that can precisely capture the requirements of HIPAA. We also need an enforcement engine that can enforce the privacy policies specified in the language. In the current work, we evaluate eXtensible Access Control Markup Language (XACML) as a candidate specification language for HIPAA privacy rules. We evaluate XACML based on the set of features required to sufficiently express HIPAA, proposed by a prior work. We also discuss which of the features necessary for expressing HIPAA are missing in XACML. We then present high level designs of how to enhance XACML's enforcement engine to support the missing features.

1 Introduction

Business organizations such as banks, hospitals, *etc.*, collect private information from their customers to perform their business functions. They often use computer information systems to manage and manipulate the collected private information of their customers. Government regulations mandate how this information can be used or disclosed by the organizations. For instance, the Health Insurance Portability and Accountability Act (HIPAA) [8] mandates how the protected health information (*phi*) of the patients collected by the health care providers can be legally used or disclosed. Intentional (or, unintentional) violations of these regulations can cause heavy financial penalties and sanctions [16]. Thus, it is desirable to have an expressive specification language for expressing these privacy regulations. Furthermore, we need to have corresponding enforcement mechanism by which privacy policies like these can be efficiently enforced.

OASIS's eXtensible Access Control Markup Language (XACML) [18] is one of the most popular access control specification languages. Along with the rich specification language, XACML [18] also has a robust enforcement engine that can enforce policies specified in the language. Although, XACML is an expressive specification language it lacks features for sufficiently specifying privacy policies like HIPAA. This is natural as XACML is designed for specifying access control policies instead of privacy policies like HIPAA. The focus of the current work is to assess XACML's adequacy for expressing HIPAA. More precisely, we describe what features a specification language requires to sufficiently specify HIPAA. We also discuss which of these necessary features XACML possesses and also propose extensions of XACML to support the missing features.

One of the apparent advantages of extending XACML to support privacy policies like HIPAA is that, one uniform specification language and enforcement mechanism can be used to specify and enforce the access control policies and the privacy policies of the system. Managing the access control policies and the privacy policies differently is cumbersome as an action can be mandated by both policies. However, if we use XACML for expressing both policies then XACML's enforcement mechanism will combine the permissibility decision of an action by using the policy combination algorithms (PCAs). Furthermore, organizations can have their own business privacy policy on top of the federal privacy regulations. In this case, the organization's privacy policy would be the composition of their business privacy policy and the federal privacy regulations. This composition of privacy policies can be very easily achieved by XACML.

Contributions. In the current work, we evaluate XACML as a possible specification language for expressing HIPAA. To the best of our knowledge, the current work is the first to consider XACML as a possible specification language for HIPAA. Our evaluation of XACML as a candidate specification language is based on a set of features required for expressing HIPAA, proposed by DeYoung *et al.* [4]. In our evaluation of XACML, we found out that XACML has some rich enough features (*e.g.*, attributes, policy/policy rule combination, *etc.*) to support HIPAA. However, it lacks some other necessary features (*e.g.*, event history, obligations, subjective belief, reference to other rules, *etc.*) to adequately capture the HIPAA privacy rules. We believe that the support for the missing features will enable XACML to specify HIPAA.

To support event history, we propose a history manager that keeps track of important events that have happened in the past and can influence the permissibility of certain disclosure or usage. To this end, we analyze the policy to manually figure out the events that are necessary to be stored in our history (a relational database). We also require that the user requesting an access (use or disclose) provides the intended purpose and also the subjective beliefs.

Roadmap. Section 2 reviews the backgrounds necessary to understand our contributions. In Section 3 we discuss the different features necessary to specify HIPAA and we discuss which of these features are missing in XACML in Section 4. In Section 5, we present the extension of XACML necessary for our purpose and its enforcement architecture. Related works are discussed in Section 6. Section 7 discusses our future work and concludes.

2 Background

In this section we briefly summarize HIPAA privacy rules and XACML.

2.1 Overview of HIPAA

We now briefly overview the Health Insurance Portability and Accountability Act (HIPAA) of 1996 also referred to as Public Law 104-191. According to the Department of Health and Human Services (HHS) the goal of the HIPAA privacy regulation is to ensure that consumers can access their health information and also to protect their information from unauthorized disclosure.

More specifically, Part 164 of HIPAA deals with the security and privacy aspect of the regulation. In this work, we primarily analyze subpart E of Part 164, which deals with protecting individually identifiable health information, covering §164.502 to §164.528. These rules precisely specify the security and privacy requirements that is applicable to "*covered entities*" with respect to protected health information (*phi*). As defined by HIPAA and the HHS, covered entities include health plans, health care clearinghouses, such as billing services and community health information systems, and health care providers that transmit health care data in a way that is regulated by HIPAA. *Protected health information* (*phi*) refers to individually identifiable health information except a few cases where such information falls under the jurisdiction of other federal regulations such as the Family Educational Rights and Privacy Act (FERPA).

The privacy rules regulate the following kinds of actions: (1) Usage of the *phi* within the covered entity itself. (2) Disclosure of *phi* to some other entity.

Furthermore, the purposes for which the covered entity (or, any other entity) is using or disclosing the *phi* is also referred in the privacy rules. The following is an incomplete list of purposes that are used: treatment; payment; health care operations; creating de-identified *phi*; communicate; marketing; reporting to public health authority; health oversight.

When disclosing *phi*, the role of the entity to which the disclosure is made is also important. An incomplete list of the different roles that are referred in the HIPAA rules include: individual (i.e., the person whose *phi* is about); representative of an individual; business associates of a covered entity; healthcare provider; an attorney representing whistleblower; group health plan; Health Maintenance Organization (HMO); public health care authority; public health or government authority authorized by law to receive child abuse report; a person who may have been exposed to a communicable disease; employer of an individual; a family member, other relative, or a close personal friend of an individual, or any other person identified by the individual; a person subject to Food and Drug Administration (FDA) regulated activity.

The HIPAA rules also often refer to other documents and contracts among the entities involved with the applicable disclosure. We briefly summarize these documents and contracts in the following discussion.

Privacy notice. According to the privacy rules, when a privacy notice is required, an access must be consistent with the privacy notice, in addition to following privacy rule (§164.502(i)). "*A covered entity that is required by §164.520 to have a notice may not use or disclose protected health information in a manner inconsistent with such notice. A covered entity that is required by §164.520(b)(1)(iii) to include a specific statement in its notice if it intends to engage in an activity listed in §164.520(b)(1)(iii)(A)-(C), may not use or disclose protected health information for such activities, unless the required statement is included in the notice.*" According to the above clause, an organization must check each access against the privacy notice. Therefore, privacy no-

tices should be encoded as policies that must also authorize a request for it to be allowed. When an organization has multiple privacy notices (for example, Google had over 70 different privacy policies before consolidating them), then it is necessary to remember for each patient which policy encodes the privacy notice for that patient and checks with that policy.

Authorizations. Accesses (use or disclose) to *phi* not explicitly authorized by the privacy rules can still be allowed when a valid authorization from the individual is obtained for the specified purpose. This is explicitly mentioned in the privacy rule §164.508 which specifies that: "*Except as otherwise permitted or required by this subchapter, a covered entity may not use or disclose protected health information without an authorization that is valid under this section. When a covered entity obtains or receives a valid authorization for its use or disclosure of protected health information, such use or disclosure must be consistent with such authorization*." To enforce this clause, authorizations signed by individuals also need to be encoded as policies and checked against.

Contracts and Restrictions. The HIPAA privacy rules in §164.522 requires that a covered entity must permit an individual to request that the covered entity restrict: (A) Uses or disclosures of protected health information about the individual to carry out treatment, payment, or health care operations; and (B) Disclosures permitted under §164.510(b). A covered entity is not required to agree to a restriction, but if it agrees to it, it must respect the restriction.

Furthermore, the HIPAA privacy rules in §164.510 allows a covered entity to use or disclose protected health information without the written consent or authorization of the individual as described by §164.506 and §164.508, respectively, provided that the individual is informed in advance of the use or disclosure and has the opportunity to agree to or prohibit or restrict the disclosure in accordance with the applicable requirements of this section. The covered entity may orally inform the individual and obtain the individual's oral agreement or objection to a use or disclosure permitted by this section.

2.2 XACML

Architecture. The main components of the XACML architecture include a Policy Enforcement Point (PEP), a Policy Decision Point (PDP), a Policy Information Point (PIP), a Policy Administration Point (PAP), and obligations service. The **PEP** performs access control, by receiving decision requests, consulting the PDP for authorization decision, and enforcing the decisions. The **PDP** evaluates applicable policies and yields authorization decisions, together with obligations and advice, if any. The **PIP** acts as a source of attribute values, such as subject, resource, action, environment attributes. The **PAP** administrates policies and policy sets and makes them available to the PDP. The **obligations service** handles obligations forwarded by the PEP. However, XACML does not specify how PIP, PAP and obligations service should behave and how they should be implemented.

```
<PolicySet> := <Target><Policy>+[Obligations]
 Attributes: PolicySetId, PolicyCombiningAlgId
<Policy> := <Target><Rule>+[Obligations]
 Attributes: PolicyId, RuleCombiningAlgId
<Rule> := [Target][Condition]
 Attributes: RuleId, Effect
```

Figure 1: XACML schema of policy set, policy and rule in BNF form

Rules, Policies, and Policy-sets. Both XACML 2.0 and 3.0 [18] define three levels of policy elements: rules, policies, and policy-sets. A *rule* is the most basic policy element; it has three main components: a *target*, a *condition*, and an *effect*. The target defines a set of subjects, resources, and actions that the rule applies to; the condition specifies restrictions on the attributes in the target and refines the applicability of the rule; the effect is either `Permit`, in which case we call the rule a *permit rule*, or `Deny`, in which case we call it a *deny rule*. If a request satisfies both the rule target and rule condition, the rule *is applicable* to the request and yields the decision specified by the effect element; otherwise, the rule *is not applicable* to the request and yields the decision `NotApplicable`.

Unlike XACML 2.0, XACML 3.0 allows one to specify obligations and advice in a rule, so the rule would return a decision together with a set of (possibly empty) obligations and advice if it is applicable to a request. Each obligation represents functions to be executed in conjunction with the enforcement of an authorization decision. *Advice* is newly added in XACML 3.0, which is a supplementary piece of information provided together with a decision, and it is like an optional obligation, which can be safely ignored by the PEP without .

A *policy* consists of four main components: a *target*, a *rule-combining algorithm (RCA)*, a set of *rules*, and *obligations/advice*. The policy target decides whether a request is applicable to the policy and it has a similar structure as the rule target. The RCA specifies how the decisions from the rules are combined to yield one decision. A *policy-set* also has four main components: a *target*, a *policy-combining algorithm (PCA)*, a set of *sub-policies*, and *obligations/advice*. A sub-policy can be either be a policy or a policy-set. The PCA specifies how the results of evaluating the sub-policies are combined to yield a decision. Figure 1 shows the schema of policy set, policy and rule in BNF form of the base XACML specification language.

Policy Combining Algorithms. XACML 2.0 and 3.0 have a number of standard RCAs and PCAs. Us-

ing "Permit-overrides", "Deny-overrides", and "First-applicable" among them are helpful and sufficient for combining HIPAA policies/rules.

3 Features for HIPAA Specification

In this section, we inventory and briefly summarize the features, proposed by DeYoung *et al.* [4, 5], that a policy specification language requires to sufficiently capture the HIPAA privacy rules.

Attributes. Each HIPAA privacy rule mandating a disclosure or usage can restrict the sender's, receiver's, the subject's, and the message's current attributes. For instance, the HIPAA privacy rule §164.502(a)(1)(i) specifies that: "*A covered entity is permitted to use or disclose protected health information as follows: To the individual*". According to this regulation (when considering a disclosure action), the sender's role attribute must be covered entity, the receiver's and the subject's role attribute must be individual, and the information in question is the subject's *phi* attribute.

Attribute Inference Policy. *Attribute inference policies* specify whether a certain individual has a specific attribute based on conditions on his current attributes. Consider the HIPAA privacy policy rule in §164.502(a)(1) that allows a covered entity to send a patient's *phi* to the patient's personal representative. While evaluating this policy rule, one might need to check whether a certain individual p_1 is the personal representative of the patient p_2. Attribute inference policies can specify under what circumstances p_1 can be considered a personal representative of p_2. An example of such an attribute inference policy can be found in §164.502(g)(2) of HIPAA. It specifies that p_1 can be considered the personal representative of p_2 when p_2 has the authority to make health care decisions for p_1 where p_1 is either an adult or an emancipated minor.

Past Events. The HIPAA privacy rules restrict a request for disclosure or usage of a patient's protected health information (*phi*) based on some events on the past. Consider the regulation §164.502(e)(1)(i) which mentions that: "*A covered entity may disclose protected health information to a business associate and may allow a business associate to create or receive protected health information on its behalf, if the covered entity obtains satisfactory assurance that the business associate will appropriately safeguard the information.*" According to this regulation, the covered entity can disclose a patient's *phi* to its business associate when it has already received satisfactory assurance from its business associate regarding the safeguarding of the *phi*.

Obligations with Deadlines. The HIPAA privacy rules also impose obligatory restrictions on the covered entity. Furthermore, the obligations have specific deadlines within which the obligation needs to be carried out. Consider the §164.524(b)(2)(i) of HIPAA, which mentions: "*the covered entity must act on a request for access no later than 30 days after the receipt of the request*". Here, the covered entity is obligated to act within 30 days after it has received a request for access from an individual.

Purpose of Usage or Disclosure. The HIPAA privacy rules restrict certain usage or disclosure requests based on the purpose of that action. For instance, the HIPAA privacy rule in §164.506(c)(1) specifies that: "*A covered entity may use or disclose protected health information for its own treatment, payment, or health care operations.*". This HIPAA privacy rule allows a covered entity to use or disclose *phi* of patient for the purpose of either its own treatment, payment, or health care.

Subjective Beliefs. The HIPAA privacy rules allows a certain disclosure or usage of a patient's *phi* based on the covered entity's subjective belief or professional judgement. An example of such a HIPAA privacy rule can be found in §164.512(f)(5) of the regulation. It states that: "*A covered entity may disclose to a law enforcement official protected health information that the covered entity believes in good faith constitutes evidence of criminal conduct that occurred on the premises of the covered entity*". This rule allows a covered entity to disclose *phi* of a patient to the police for reporting a crime on premise and additional believes the *phi* can be used as evidence.

Reference to Other Laws/Rules. The HIPAA privacy rules also restrict a certain disclosure or usage of a patient's *phi* based on other laws and also others rules (sections and paragraphs) of HIPAA. One example of the HIPAA privacy rule referring to another law can be found in §164.512(a)(1) which specifies that: " *A covered entity may use or disclose protected health information to the extent that such use or disclosure is required by law and the use or disclosure complies with and is limited to the relevant requirements of such law*". The HIPAA privacy rule §164.502(a)(1)(ii) can serve as an example where one rule of HIPAA refers to another HIPAA privacy rule. It specifies that: "*A covered entity is permitted to use or disclose protected health information as follows: For treatment, payment, or health care operations, as permitted by and in compliance with §164.506;*".

Policy/Policy Rule Combination. HIPAA has different types of privacy rules based on the restrictions they impose. More precisely, (i) some of the HIPAA privacy rules allow certain disclosure, (ii) some of the HIPAA privacy rules prohibit certain disclosure to take place, (iii) some of the rules allow certain disclosure when certain condition is satisfied, and (iv) some of the rules require certain disclosure to take place. To check whether certain disclosure or usage is in compliant with the HIPAA privacy rules, one should be able to consult all the above types of privacy rules to get decisions and

combine them to get one consistent decision. An example of type (i) rules can be found in §164.506(c)(2) of HIPAA, which specifies that: "*A covered entity **may** disclose protected health information for treatment activities of a health care provider*". The HIPAA privacy rule §164.502(g)(3)(ii)(B) demonstrates the type (ii) rules. It specifies that: "*If, and to the extent, prohibited by an applicable provision of State or other law, including applicable case law, a covered entity **may not** disclose, or provide access in accordance with §164.524 to, protected health information about an unemancipated minor to a parent, guardian, or other person acting in loco parentis; and...*". An example of type (iii) privacy rules can be found in §164.508(a)(1) of HIPAA which specifies that: "*Notwithstanding any provision of this subpart, other than the transition provisions in §164.532, a covered entity **must** obtain an authorization for any use or disclosure of psychotherapy notes,...*". Example of type (iv) rule can be found in the HIPAA privacy rule §164.502(a)(2)(ii) which specifies that, "*A covered entity **is required to** disclose protected health information: When required by the Secretary under subpart C of part 160 of this subchapter to investigate or determine the covered entity's compliance with this subpart.*".

4 Evaluating XACML for HIPAA

We identify the mismatches that occur when expressing HIPAA in XACML. These serve as the motivation for future research and development of access control specification languages. We use XACML as an example of state-of-the-art access control language with enforcement support.

Stateful Policies vs. Stateless Mechanism

XACML policies are largely stateless. Essentially XACML provides a component that takes a request as input, and returns a decision. The XACML architecture puts forward keywords such as PEP, PDP, PAP, and PIP. However, it does not suggest how to implement each of PAP and PIP, let alone modeling their interactions.

When using XACML to encode a set of complicated policies, all one can do is to create a stateless policy that takes requests as inputs and give decisions as outputs. Anything else is beyond the actual XACML standard. The HIPAA privacy rules, however, goes beyond a simple policy providing answers to requests.

Obligations. Although XACML seems to integrate obligations as part of it, it treats obligations largely as black boxes, without specifying what an obligation should include and how to handle them. In short, XACML does not assign any semantics to obligations, which we believe is necessary.

Event History. XACML is stateless and assumes any stateful information (e.g, history) is kept outside the policy engine. One possibility is to use Condition semantics of XACML 3.0 to handle history. However, one has to assume that there exists a sophisticated component outside the policy engine that maintains relevant history information and knows exactly which part of the history information is needed for a given request to put such information in the request context. In a sense, one has to assume a policy engine beyond XACML to handle these things. We believe that the decisions about how history information are maintained and used are largely policy driven, and should be handled together with access control policies, inside the XACML framework.

Policy-Directed Attribute Retrieval. In HIPAA, different attributes need to be provided for different requests. Deciding which attributes to retrieve, is often dictated by the policy itself. XACML currently does not contain support for this operation.

Policy-Directed Policy Retrieval. As we have discussed before, the HIPAA privacy rules can refer to other documents or contracts (*e.g.*, privacy notice, authorization, *etc.*) between the covered entity and the subject in question. If such a document or contract exists, it can dominate the response from the regular privacy rules. In that sense, we can consider these documents or contracts as separate policies and retrieve them when necessary. Currently, XACML does not support such interactions.

Interactive vs. Non-interactive Policy Evaluation

Reading HIPAA, one gets the sense that policy evaluation needs to be more interactive. For a disclosure request, depending on which justification one plans to use for the disclosure, a different set of conditions need to be checked. One cannot simply send a request and get back a decision.

Purpose of Disclosure or Usage and Subject Beliefs. The HIPAA regulation sometimes permits a disclosure or usage of a patient's *phi* based on the purpose or based on the subject's belief. However, deciding whether certain disclosure or usage is requested for certain purposes, is difficult. It is impossible to decide from the static context of the request arguments. The same is true for subjective beliefs. It is often difficult to decide subjective beliefs without interacting with the requester.

Reference to Other Laws/Rules. As we have seen before, HIPAA privacy rules can refer to other HIPAA privacy rules and also other laws. As a result of which, while evaluating a privacy rule we might have to evaluate a different rule (referred in the original privacy rule) first before making decision about the first rule. Currently, XACML does not support such interactions between policy/policy rules.

Attribute Inference vs. Authorization Decisions

In XACML, all rules assign some kind of truth value to a particular request, and one cannot write a rule/policy assigning truth value to a query that is not an access request. For example, in HIPAA one condition for accessing *phi* is that the requester is a personal representative of the patient. However, HIPAA has guidelines that dictate whether someone should be considered to be a personal representative. Ideally these conditions for deciding personal representative should also be specified as XACML policies and rules. However, these policies and rules are not about deciding the request, but about the inference of some attribute relevant to the current decision. Therefore, they cannot be expressed in current XACML.

Quantification Over Infinite Domains

As pointed out by existing work [2, 4–6], concise specification of the HIPAA privacy rules require quantifications over the infinite domains of the involved principals, message attributes, messages, *etc.* XACML supports implicit universal quantification (outer-most) of the sender, receiver, subject, message, message attributes, *etc.*, of the use or disclosure action which the rule mandates. However, while specifying HIPAA privacy rules, the condition associated with a privacy rule can also have quantifications. Currently, XACML does not support the specification of the explicit quantifications appearing in the condition of a rule.

5 Extensions of XACML to Support HIPAA Policies

Inspired by Barth *et al.* [2], we divide the HIPAA privacy regulation regarding disclosure or usage of a patient's *phi* into two types of privacy rules, *allowing policy rules* and *prohibitive policy rules*. An allowing policy rule (*e.g.*, §164.502(a)(1)(i), *etc.*) enables a disclosure or usage whereas a prohibitive policy rule (*e.g.*, §164.508(a)(2), *etc.*) permits a disclosure only when its associate condition is satisfied. We describe how these rules are combined in section 5.6. Each HIPAA policy rule regulating a disclosure or usage contains the following restrictions: (1) sender's attributes, (2) recipient's attribute, (3) subject's attribute (the individual whose *phi* is considered), (4) purpose of the disclosure, (5) the information that is being disclosed (*e.g.*, age, name, ssn, *etc.*), (6) obligations, (7) history, and (8) other conditions. In this section, we summarize how each of these are specified in the extended XACML. Our proposed extensions are based on the investigation of the HIPAA privacy rules in §164.502-§164.514, §164.522, and §164.524. Note that, the goal of this work is not completely specifying and enforcing the HIPAA privacy rules rather evaluating XACML as a candidate for specifying and enforcing HIPAA privacy rules.

Assumptions. We now discuss the assumptions we make while considering XACML as a specification language for HIPAA. In our system, the actions of the system that are regulated by the privacy policy are disclose (*e.g.*, send a message, send postal mail, *etc.*), use (*e.g.*, read, write, *etc.*), request (*e.g.*, request from patient, *etc.*), and access (*e.g.*, patient accessing her own *phi*, *etc.*). More precisely, we only regulate disclosure or usage regarding to the *phi* of a patient. We also assume that it is the responsibility of the sending user to tag the message with the appropriate attributes (*e.g.*, address, ssn, age, *etc.*) based on the content of the message. Additionally, current work makes the assumption that when an obligation is incurred, the user incurring the obligation (*obligatee*) will be permitted according to the privacy and access control policy. However, this might not be the case. Relaxing this assumption and designing a static analysis of the policy to check whether it has this desired property is a subject of future work.

In the current work, we have abstracted away some portions of HIPAA. Consider the regulation in §164.502(g)(3)(ii)(A), that allows a covered entity to disclose the *phi* to the guardian provided that other laws allow it. It is not feasible to encode all possible applicable laws in our language. As a result, we use an oracle (possibly a company attorney) to decide whether the disclosure is allowed by other laws. We additionally assume that the patient policies that the covered entity agrees to comply with, is consistent with the HIPAA privacy rules.

5.1 Obligations

Sometimes HIPAA policies specify obligations required to be performed by covered entities. For example, the HIPAA regulation §164.524(b)(2)(i) says that the covered entity is obligated to act within 30 days after it has received a request from an individual. As mentioned before, XACML's support for obligations is not rich enough to capture the obligatory requirements of HIPAA. As a result, in the current work, we adopt the obligation model by Li *et al.* [12] to support specification and enforcement of obligations in XACML. Note that, there are other approaches to manage obligations [3, 6, 15], but the model by Li *et al.* [12] is a natural fit as it can readily be used with XACML without any significant modifications. The key ideas of the state-machine-based approach proposed in [12] are as follows. An obligation is modeled as a state machine that communicates with the PEP using events. The PEP manages the life-cycle of obligations. An obligation includes rulesets to specify its responses to input events. These responses include changing its state in response to events, which

informs the PEP about what course of actions it should take regarding the request, and generating events, which inform the environment about what actions must be taken to fulfill the obligation. Some of these actions are deployment specific. These deployment specific actions are implemented by obligation modules. Multiple obligation modules can be attached to the PEP, each implementing some actions. These obligation modules communicate with the PEP and the obligations through an event interface. The details of this approach can be referred to [12].

5.2 History Management

The HIPAA privacy regulation sometimes allows a certain disclosure or usage of a patient's *phi* when certain condition/event in the past (temporal condition) is true. To facilitate this, we propose a *history manager* that keeps track of important past events that might influence the permissibility of a certain disclosure or usage. A history manager is a relational database that saves important events and can be queried efficiently. In the example above, whenever we receive a court order requesting the *phi* of a certain patient, we would save this event on the history manager. Now, in response to the court order, if the covered entity attempts to send the required information to the court, we check whether the covered entity actually received a court order. We achieve this by checking history table for an entry which is a court order that the covered entity received. Note that, design of such history manager has been proposed in the literature [6, 10] but we design it specifically for XACML and discuss its interactions with PDP to make an access decision.

Recall that, the history condition of a policy rule can contain quantifications over the domains of principal, message, and message attribute. XACML cannot express such quantifications. We overcome this by expressing the history conditions as stored database procedures which take arguments. In the rule specification, we refer to the appropriate database procedure. We use events to pass the proper arguments of the database procedure. We follow the same approach for quantification in other elements of the conditions (*e.g.*, attribute inference, *etc.*). Now, to support event history in the condition of the privacy rules, we have to extend the *<Policy>* element of XACML which we discuss just below.

5.3 Interactions with Users

The HIPAA regulation sometimes permits a disclosure or usage of a patient's *phi* based on the purpose or based on the subject's belief. However, deciding whether certain disclosure or usage is requested for certain purposes, is difficult. It is impossible to decide from the static context of the request arguments. We follow the approach of Lam *et al.* [11] and require that the user provides the purpose as an argument of the request. We present a list of possible purposes to the user and she chooses the appropriate one. Automatically determining whether a certain action is for some certain purpose [17] is out of the scope.

It is also not trivial how to model subjective belief of a principal in a computer information system. Thus, whenever we try to evaluate a policy rule that allows a usage or disclosure based on a subject's belief, we require additional information from the principal requesting the action. The additional information in this case is the information about the subjective belief.

Extension of the *<Policy>* element. In order to support interactions with users during policy evaluations, the *<Policy>* element in XACML needs to be extended. As aforementioned, some attribute values like subjective beliefs might be missing when checking whether a condition is satisfied, and thus user inputs might be required. In this case, the policy evaluation has to be stopped, and events should be sent out to inform users to provide the missing information. And then user inputs, if provided, will be sent back also by events. One possible extension is that attributes that will be required during the policy evaluation are specified in an optional *<RequiredAttributeList>* element. Hence we extend the *<Policy>* element in XACML, as shown below.

```
<Policy> := [RequiredAttributeList]<Target><Rule>+
            [Obligations]
 Attributes: PolicyId, RuleCombiningAlgId
<RequiredAttributeList> := <RequiredAttributeSelector>+
<RequiredAttributeSelector> := [Keys]
 Attributes: AttributeId, DataType, Source, DatabaseId
(optional), TableId (optional)
<Source> := "User"|"Database"|"Oracle"
<Keys> := <Key>+
<Key> := <KeyValue>
 Attributes: KeyId
```

The *Source* of a *<RequiredAttributeSelector>* element can be *User*, *Database* or *Oracle*, which indicates where the required attribute comes from. If the attribute is from the database, *DatabaseId*, *TableId*, and *Keys* of the table should be specified for the query.

According to the *<RequiredAttributeList>* element, the system will send events to users (i.e., requesters) informing what missing attribute values are required, query the database to retrieve the attribute values, or query the oracle for additional information (e.g., whether a disclosure is allowed by other laws). (Note that, for condition of the rules containing history restrictions we have to query the history database.) Once responses are obtained from the users or the database, events carrying information about the required attributes will be sent back to the policy. Therefore, there should exist a way to get the attribute values from the incoming events when checking *<Condition>* in policy rules. Hence an *<EventSelector>* element is added into the *<Expression>* element substitution group.

<Condition> := <Expression>
The <Expression> element substitution group includes:
<AttributeSelector>, <AttributeValue>,
<VariableReference>, <ActionAttributeDesignator>,
<ResourceAttributeDesignator>, <Function>,
<SubjectAttributeDesignator>, <Apply>,
<EnvironmentAttributeDesignator>, <**EventSelector**>
<EventSelector> :=
Attributes: EventType, EventField, DataType

5.4 Attribute Inference Policies

Attribute inference policies specify whether a certain individual has a specific attribute based on conditions on his current attributes. Thus, an attribute inference policy can be viewed as an *Oracle* which responds with *True* or *False* to queries like "Does user p_1 has attribute a_1 based on p_1's current attributes?". Note that, in the context of distributed authorization Li *et al.* [13] proposed the RT language which achieves something similar to what we propose. However, we propose attribute inference policies in context of HIPAA and XACML.

To facilitate attribute inference policies, *<Condition>* is further extended in a way that *<AttributeInferencePolicyReference>* is added into the substitution group of *<Expression>* to support references to attribute inference policies.

<Condition> := <Expression>
The <Expression> element substitution group includes:
<AttributeSelector>, <AttributeValue>,
<VariableReference>, <ActionAttributeDesignator>,
<ResourceAttributeDesignator>, <Function>,
<SubjectAttributeDesignator>, <Apply>,
<EnvironmentAttributeDesignator>, <**EventSelector**> ,
<**AttributeInferencePolicyReference**>
<AttributeInferencePolicyReference> := <Input>$^+$
Attributes: AttributeInferencePolicyId

The same schema for HIPAA privacy policies provided by the extended XACML can be reused to specify this type of attribute inference policies. The only small difference is that the evaluation results of these policies are *True* or *False*, instead of *Permit* or *Deny*.

5.5 Additional Policies

An organization that is interested in enforcing HIPAA will have some additional policies (*i.e.*, organizational access control policies and patient policies). We present these here and show how they fit the big picture of enforcing HIPAA. In section 5.6, we present how these policies are combined with the HIPAA policies.

Organizational Access Control Policies. The HIPAA privacy regulation mandates what information of a patient can the covered entity disclose or use and under what circumstances. However, the covered entity (*e.g.*, hospital, clinic, doctor's office, *etc.*) might have some additional access control requirements that further restricts which employees of the covered entity can access the *phi* of a certain patient. For instance, a covered entity might only allow the assigned doctors and the assigned nurses to access the *phi* of a certain patient. Thus, even in the case where HIPAA allows doctors/nurses to use or disclose the *phi* of a patient, they will be denied in the case if they are not the patient's assigned doctor/nurse.

Patient Policies. *Patient policies* are those policies specified by the patients themselves. According to §164.522 of HIPAA, a covered entity can agree or disagree to comply with the patient policy. If the covered entity agrees to do so, the covered entity must comply with the patient policy along with the HIPAA policies.

5.6 Policy Combination

HIPAA policy combination. The HIPAA policies can be organized in the following way. In the top level, permit-overrides PCA is used to combine two types of policies: (1) required policies that specify disclosures that are required and must be permitted, such as §164.502(a)(2)(i); and (2) permitted policies that specify uses or disclosures that might be permitted, such as §164.502(a)(1)(i). All required policies are combined using permit-overrides PCA, while the permitted policies are combined using deny-overrides PCA. The permitted policies are further divided into allowed policies, which are combined with permit-overrides PCA, and prohibitive policies, which are combined using deny-overrides PCA. Permit-overrides PCA and deny-overrides PCA are used in most cases, and sometimes their ordered versions are utilized. Thus the existing PCAs in XACML are sufficient to combine results of policy evaluations.

Combining additional policies. A disclosure or usage request of a covered entity is permitted, when it is allowed by all the policies: the organizational access control policy, the patient policy (if there is one), and the HIPAA policies. These policies are combined using the ordered-deny-overrides PCA. Recall that, ordered-deny-overrides PCA is the same as deny-overrides PCA, except that policies have to be evaluated exactly in the order they appear. Policies can be arranged in the following order: the access control policy, the patient policy, and the HIPAA policies. For performance sake, once a policy denies a request we do not evaluate the policies ordered after it and simply deny the request.

5.7 Architecture Design

Based on XACML's architecture and the architecture presented in [12], we propose an architecture which supports the enforcement of policies specified in our extended XACML language. Figure 2 shows our proposed architecture. The overall system is divided into two parts: a HIPAA Compliance Checking Component (HCCC), and an External Environment (EE). The EE is where applications are executed (e.g., a web-based HIPAA in-

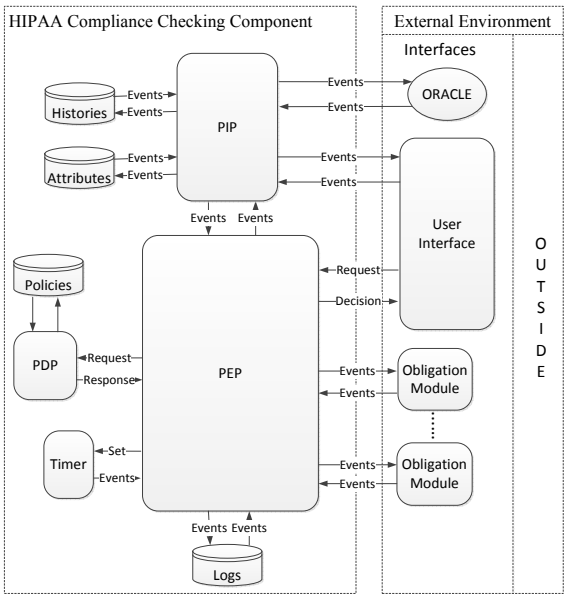

Figure 2: Proposed architecture

formation system), and the HCCC helps the EE decide whether a usage or disclosure of *phi* is permitted.

The main components in HCCC include a PEP, a PDP, a PIP, databases storing policies, attributes, histories, and logs, respectively, an ORACLE, and a Timer. **PEP** receives requests, consults the PDP for a decision, handles any associated obligations, and makes the final decision about the request. The **PDP** evaluates the attribute-based policy provided by the policy database, and returns, to the PEP, a PDP decision, together with obligations, if any. **PIP** serves as an attribute query point for subjects, resource and environmental attributes, and an information query point which obtains inputs from users, histories from the database and additional information from the Oracle. **Attribute database** acts as a storage of subject and object attributes needed in policy evaluations. **History database** stores history records such as authorizations or court orders for covered entities. **Policy database** stores HIPAA privacy policies, access control policies, and patient policies. **Log database** can be used to keep logs, which leaves a room for auditing in the future work. The **Oracle** interacts with the PIP in a way that the PIP queries the Oracle and gets back a "Yes/No" (boolean) response. This is necessary for capturing whether a use or disclosure is allowed by other laws. The **Timer** informs the PEP that either a specific time point (e.g., 11:59P.M. on January 27th, 2012; it is pre-set by the PEP) arrives or a time duration (e.g., 5 minutes; it is pre-set by the PEP) is up.

The EE interacts with the HCCC through an interface, which include a user interface and zero or more Obligation Modules. **User interface** is where users can submit their requests and obtain whether the requests are allowed or denied according to HIPAA privacy policies and other policies. Besides, it will interact with the PIP via events if user inputs are required during policy evaluations. **Obligation modules** implement obligation-handling functionalities (such as notifying users, and writing to logs). More details can be referred to [12]. For XACML to support attribute inference policies, it needs to be changed so that the policy engine should support the following feature: while evaluating the original access request, the policy generates another request (representing a question about some attribute) and selects and evaluates other relevant policies for this new request (for attribute inference), and then the attribute inference result will be integrated with other policy to make a decision about the original request.

6 Related work

May *et al.* [14] present a formalism, based on HRU access control matrix model [7], called Privacy APIs to encode HIPAA [8]. They only translate §164.506 of the 2000 and 2003 version of HIPAA in their formalism and use the SPIN model checker [9] to check different desired invariants of the specification. Roughly, their work concentrates on specifying privacy policy and its analysis. Our work on the other hand tries to address the adequacy of XACML to specify and to enforce HIPAA. In that sense, their work is complimentary to ours.

Lam *et al.* [11] propose a formalism, pLogic, based on a specific fragment of stratified Datalog with one alternation of negation. They consider the subparts §164.502, §164.506, and §164.508 of HIPAA. Their specification language does not support obligations. The goal of the work of Lam *et al.* [11] to present a specification language and enforcement mechanism for HIPAA. The goal of our work is on the contrary is to evaluate XACML as a candidate for specifying and enforcing HIPAA.

Garg *et al.*, [6] propose a privacy policy specification language based on first order logic. Their specification language [4, 5] can completely capture the HIPAA privacy rules. They also present an auditing algorithm that can detect violations of policies specified in their specification language. They argue that auditing is necessary for the enforcement of HIPAA as some of the information (*e.g.*, reference to other laws, subjective belief, *etc.*) necessary for making decisions cannot be obtained in runtime. In our work, we assume that the user provides the necessary information about the purpose and the subjective beliefs. When the same assumption is made, the auditing algorithm of Garg *et al.*, [6] can be used in an online fashion to enforce HIPAA. Recall that, the goal of the current work is not to design a specification language and its associated enforcement engine to precisely specify and to enforce the HIPAA privacy rules. The goal

of our work on the contrary is to evaluate XACML's adequacy to specify and to enforce HIPAA. In that sense, their work is complimentary to ours.

The Enterprise Privacy Authorization Language (EPAL) [1] is designed to specify organizational privacy policies. EPAL supports purpose of a disclosure or usage and also allows obligations to be incurred when an action is either allowed or denied. However, in EPAL's policy rules there is no support for restricting the attributes of the sender of a information and also the subject of the information. Furthermore, they do not support past temporal conditions to be added to the condition of a rule.

7 Conclusion

In this work, we evaluate XACML based on the features proposed by DeYoung *et al.* [4, 5] and found out that XACML has some of the necessary features but lacks some other features. We then present high level designs to extend the specification language and enforcement architecture to support the missing features.

Currently, we do not have a prototype of our approach. In on-going work, we are extending XACML enforcement engine to support the extensions we have proposed for the specification of HIPAA.

8 Acknowledgement

We would like to thank Dr. Anupam Datta and the anonymous reviewers for their helpful suggestions. The work reported in this paper was partially supported by the Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265, NSA-CMU Grant No. 1130145-280847, and by the National Science Foundation under Grant No. 0905442 and 0964710.

References

[1] Enterprise privacy authorization language (EPAL) version 1.2, Nov. 2003. http://www.zurich.ibm.com/pri/projects/epal.html.

[2] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. *IEEE Symposium on Security and Privacy*, pages 184–198, 2006.

[3] D. Basin, F. Klaedtke, and S. Müller. Policy monitoring in first-order temporal logic. In *Proceedings of the Computer Aided Verification (CAV'10)*.

[4] H. DeYoung, D. Garg, L. Jia, D. Kaynar, and A. Datta. Experiences in the logical specification of the hipaa and glba privacy laws. In *Proceedings of the WPES'10*, New York, NY, USA. ACM.

[5] H. DeYoung, D. Garg, L. Jia, D. K. Kaynar, and A. Datta. Technical report CMU-CyLab-10-007: Experiences in the logical specification of the HIPAA and GLBA privacy laws, 2010.

[6] D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: theory, implementation and applications. In *Proceedings of the 18th ACM CCS'11*, New York, NY, USA. ACM.

[7] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, Aug. 1976.

[8] Health Resources and Services Administration. Health insurance portability and accountability act, 1996. Public Law 104-191.

[9] G. Holzmann. *Spin model checker, the: primer and reference manual*. Addison-Wesley Professional, first edition, 2003.

[10] K. Krukow, M. Nielsen, and V. Sassone. A logical framework for history-based access control and reputation systems. *J. Comput. Secur.*, 16(1):63–101, 2008.

[11] P. E. Lam, J. C. Mitchell, and S. Sundaram. A formalization of hipaa for a medical messaging system. In *Proceedings of the TrustBus '09*, Berlin, Heidelberg. Springer-Verlag.

[12] N. Li, H. Chen, and E. Bertino. On practical specification and enforcement of obligations. In *Proceedings of the CODASPY'12*.

[13] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2002.

[14] M. J. May, C. A. Gunter, and I. Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *CSFW '06*, Washington, DC, USA, 2006. IEEE Computer Society.

[15] M. Pontual, O. Chowdhury, W. Winsborough, T. Yu, and K. Irwin. Toward Practical Authorization Dependent User Obligation Systems. In *Proceedings of the 5th ASIACCS*, 2010.

[16] P. Roberts. HIPAA Bares Its Teeth: $4.3m Fine For Privacy Violation. Available at https://threatpost.com/en_us/blogs/hipaa-bares-its-teeth-43m-fine-privacy-violation-022311.

[17] M. Tschantz, A. Datta, and J. Wing. Formalizing and enforcing purpose restrictions in privacy policies. In *Proceedings of 33rd IEEE Symposium on Security and Privacy*. IEEE, 2012.

[18] XACML TC. Oasis extensible access control markup language (xacml). *http://www.oasis-open.org/committees/xacml/*.